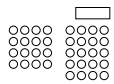
Programiranje 2: 1. pisni izpit

2. april 2025

Čas reševanja je 60 minut. Veliko uspeha!



Sedež (P.01)

Ime in priimek

Vpisna številka

1. naloga (10 točk)

Za vsakega izmed spodnjih programov prikažite vse spremembe sklada in kopice, če poženemo funkcijo `main`. Za vsako spremembo označite, po kateri vrstici v kodi se zgodi.

a)

```
1  fn main() {
2      let mut vec = vec![1, 2, 3];
3      for i in &vec {
4          println!("{}", i);
5      }
6      vec.push(4);
7  }
```

b)

```
1  fn f(c: usize) -> usize {
2      println!("{c}");
3      c * 10
4  }
5
6  fn main() {
7      let x = String::from("Hello");
8      let n = f(x.len());
9      let y = x;
10     println!("{} {}", y, n);
11 }
```

c)

```
1  fn f(x: usize, s: String) -> usize {
2      let y = x + s.len();
3      g(y)
4  }
5
6  fn g(z: usize) -> usize {
7      z * 2
8  }
9
10 fn main() {
11     let s = String::from("Hello");
12     let result = f(5, s);
13     println!("{}", result);
14 }
```

d)

```
fn f(s: &String) {
    let c = s.clone();
    println!("Copy: {}", c);
}

fn g(mut s: String) -> String {
    s.push_str(" World");
    f(&s);
    s
}

fn h() -> String {
    let wrapped = String::from("Hello");
    g(wrapped)
}

fn main() {
    let result = h();
    println!("Final String: {}", result);
}

```

2. naloga (20 točk)

Definirajmo tip `Ali<T, U>`, kot najbolj enostaven vsotni tip. Dopolnite signaturo spodnje implementacije pri čemer naj bodo tipi čim bolj splošni (morebitne odločitve utemeljite). Če v dani prostor ni treba dopisati ničesar, ga prečrtajte.

```
enum Ali<T, U> {
    Levi(T),
    Desni(U),
}

impl<T, U> Ali<T, U> {
    fn je_levi(_____ self) -> _____ bool {
        // Preveri, ali smo v varianti `levo`
    }

    fn zamenjaj(_____ self) -> _____ Ali<_____, _____> {
        // Zamenja varianti.
    }

    // Ena od teh, obe sta ok
    fn dobi_desnega(_____ self) -> _____ Option<__________> {
        // Vrne vrednost v desni varianti (če je to mogoče).
    }

    fn map_levi<F, V>(_____ self, f: _____ F) -> __________
      where
        F: __________,
    {
        // Levo vrednost preslika s funkcijo `f`
    }

    fn zamenjaj_levega(_____ self, t: _____ T) {
        // Na mestu zamenja levo varianto
    }

    fn uporabi<F, G, V, Z>(self, f: F, g: G) -> _____ Ali<V, Z>
      where
        F: __________ -> _____ Ali<V, Z>,
        G: __________ -> _____ Ali<V, Z>,
    {
        // Uporabi ustrezno funkcijo na ustrezni varianti
        // let a: Ali<usize, usize> = Ali::Desni(5);
        // let f1 = |x| Ali::Levi(x + 1);
        // let f2 = |x| Ali::Desni(vec![0;x]);
        // let k = a.uporabi(f1, f2);
        // println!("{:?}", k.dobi_desnega()); // Some([0,0,0,0])
    }
}
```

3. naloga (20 točk)

Za vsakega izmed spodnjih programov:

1. razložite, zakaj in s kakšnim namenom Rust program zavrne;
2. program popravite tako, da bo veljaven in bo učinkovito dosegel prvotni namen.

a)

```
fn print_length(s: String) {
    println!("Length: {}", s.len());
}
fn main() {
    let text = String::from("Rust");
    print_length(text);
    println!("{}", text);
}
```

b)

```
fn main() {
    let mut data = vec![1, 2, 3];
    let first = &data[0];
    data.push(4);
    println!("{}", first);
}
```

c)

```
fn main() {
    let s = String::from("hello");
    let r = &s;
    let t = s;
    println!("{}", r);
}
```

d)

```
fn f() -> &String {
    let s = String::from("hello");
    &s
}

fn main() {
    let r = f();
    println!("{}", r);
}
```

e)

```
struct O { x: Vec<i32>, }
impl O {
    fn a(&self) -> &i32 { &self.x[0] }
    fn b(self) -> Vec<i32> { self.x }
}
// Cilj je izpisati vsebino vektorja in prvi element kar se da učinkovito.
// Smiselen popravek je lahko tudi v definiciji ali implementaciji strukture O.
fn main() {
    let o = O { x: vec![1, 2, 3] };
    let first = o.a();
    println!("{:?}", o.b());
    println!("{}", first);
}
```